

[illegible]

++

Facility: STARTUP, Procedure for Starting Up a System

Abstract: This procedure is invoked by the STARTUP batch process during the startup of a VMS system. It must perform all mandatory startup activities, and invoke the site-specific startup procedure to perform optional ones.

This procedure can also be invoked by the system manager to restart any of the standard system processes, should they die for some reason.

P1 A request code.

CACHE_SERVER	Restart cluster file cache server.
CSP	Restart cluster server process.
CONFIGURE	Restart dynamic configurer.
ERRFMT	Restart error formatter.
FULL	Perform a full system startup.
JOBCTL	Restart job controller.
MINIMUM	Perform a minimal startup.
OPCOM	Restart oper. communicator.
RCP	Restart recovery control process
UPGRADE	Perform an upgrade startup.

P2 Should verification be enabled?
P8 Reserved for the customer.

Environment: No particular environment is assumed.

Author: Paul C. Anagnostopoulos

Creation: 20 April 1983

Modified by:

V04-033	BLS0349	Benn Schreiber	30-AUG-1984
	Upcase all logicals in f\$trnlm calls, and convert f\$logicals to f\$trnlm.		
V04-032	BLS0339	Benn Schreiber	10-AUG-1984
	Set OPA0 nodisconnect, nodialup and nomodem.		
V04-031	ACG0438	Andrew C. Goldstein,	3-Aug-1984 16:47
	Add startup of the file cache server process		
V04-030	KDM0105	Kathleen D. Morse	30-Jul-1984
	Print MicroVMS starting message only if P1 is null.		
V04-029	BLSC329	Benn Schreiber	30-JUN-1984
	Invoke UVSTARTUP earlier. Give opcom bigger extent.		
V04-028	BLS0325	Benn Schreiber	24-JUN-1984
	Use FULLDEVNAM when obtaining system device name. Define SYS\$SCRATCH for SYSTARTUP and equate to SYS\$MANAGER:.		
V04-027	KDM0104	Kathleen D. Morse	18-May-1984
	Fix references to UVSTARTUP.COM.		
V04-026	KDM0102	Kathleen D. Morse	14-May-1984
	Do not start ERRFMT or OPCOM on full MicroVMS boot. Only start them on MicroVMS systems, if they are explicitly specified as the P1 parameter. Do not start CSP if the P1 parameter is set to MINIMUM. Do not start CONFIGURE		

if no CI port is found.

V04-025 BLS0315 Benn Schreiber 7-MAY-1984
Ensure default set to SYSSYSROOT:[SYSEXEC]. Reset it
if we exit.

V04-024 CWH4024 CW Hobbs 07-May-1984
Bump AST and BUFIO quotas for OPCOM, to improve
real-time performance.

V04-023 WHM0004 Bill Matthews 02-May-1984
Remove reference to USESYSPARAMS getsyi item.

V04-022 WHM0003 Bill Matthews 04-Apr-1984
Invoke SYSGEN to do a write current if a separate system
parameter file is being used and modifications were made
in a conversational boot.
Make the definition of SYSSCOMMON the same as SYSSSYSROOT
when there is no common root.
Don't invoke SYSGEN to do an autoconfigure if the SYSGEN
parameter NOAUTOCONFIG is true.
Added privileges BYPASS and SHARE to CONFIGURE so it can
report errors to the operator console.
Added privilege SETPRV to OPCOM.

V04-021 ADE0001 Alan D. Eldridge 2-Apr-1983
Install images before starting system processes.

V04-020 TMK0003 Todd M. Katz 22-Mar-1983
Create the shareable logical name LNM\$DCL_LOGICAL with the
translation LNM\$FILE_DEV instead of the translations
LNM\$PROCESS,LNM\$JOB,LNM\$GROUP,LNM\$SYSTEM.

V04-019 LEB001 Linda Benson 13-MAR-1984
Add entry to run SMGMAPTRM needed by RTL SMG\$ facility.

V04-018 BLS0274 Benn Schreiber 21-FEB-1984
If uvms system, try to invoke UVSTARTUP.COM

V04-017 WHM002 Bill Matthews 13-Feb-1984
Add terminal and concealed attributes to the logical name
sys\$sysroot.

V04-016 WHM001 Bill Matthews 01-Feb-1984
Modify logical name definitions to support a shared common
system root.

V04-015 LY0453 Larry Yetto 13-JAN-1984 11:26
Comment out all CJF related code

V04-014 LY0449 Larry Yetto 5-JAN-1984 13:01
Separate the starting of the default JNLACP from the JCP INIT
command.

V04-013 BLS0248 Benn Schreiber 5-Dec-1983
Speed up, at the sacrifice of elegance.

V04-012 TMK0002 Todd M. Katz 22-Oct-1983
Create the shareable logical name LNM\$DCL_LOGICAL with the
translations LNM\$PROCESS,LNM\$JOB,LNM\$GROUP,LNM\$SYSTEM instead
of the translations LNM\$PROCESS,LNM\$GROUP,LNM\$SYSTEM.

V04-011 TMK0001 Todd M. Katz 18-Oct-1983

F 8

Create the shareable logical name LNM\$DCL_LOGICAL which is used by DCL as the logical name table name when it translates logical names.

V04-010 PRB0276 Paul Beck 27-Sep-1983 10:01
 Fix startup of CSP so it only starts if in a cluster.
 Fix loading of RUF services.
 Separate out startup of RCP so it can be restarted using STARTUP.
 Re-enable JCP INITIALIZE command to create default JNLACP.

V04-009 KFH0001 Ken Henderson 12 Sep 1983
 Fix testing of cluster member and loading of CJF.
 Get rid of 'sysgen_error' routine.

V04-008 KDM0078 Kathleen D. Morse 30-Aug-1983
 Fix SCS_EXISTS line as \$GETSYI does not return FALSE.

V04-007 KDM0077 Kathleen D. Morse 29-Aug-1983
 Add NETMBX privilege to OPCOM startup.

V04-006 CWH3002 CW Hobbs 25-AUG-1983
 Change OPCOM startup, send /ERROR to SYS\$MANGER:comp_ERROR.LOG

V04-005 PCA1028 Paul C. Anagnostopoulos 19-Aug-1983
 Reserve P8 for the customer and pass all parameters to SYCONFIG and SYSTARTUP.

V04-004 WMC0001 Wayne Cardoza 08-Aug-1983
 SCS_EXISTS is returned as a string.

V04-004 WMC0001 Wayne Cardoza 06-Aug-1983
 Make a bunch of logical names exec-mode.
 Remove JCP initialize command since it is unimplemented.
 Fix string for SCS_EXISTS.

V04-003 CWH3001 CW Hobbs 21-JUL-1983
 Add CSP (Cluster Server Process) startup.

V04-002 Paul Beck 29-JUL-1983 18:30
 Add startup code for Common Journaling Facility

V04-001 Ron Schaefer Use new logical name services.

```

saved_verify = 'f$verify(0)    ! This must be the first DCL command.
set noon
say := write sys$output
sysgen := $sysgen
cpu = f$getsyi("CPU")
microvax = ((cpu .eq. 7) .or. (cpu .eq. 8))

```

Enable all message parts so that error messages are more obvious.

```

saved_msg = f$environment("MESSAGE")
saved_default = f$environment("DEFAULT")
set message/facility/severity/identification/text
set default sys$sysroot:[sysexe]

```

If no parameters were specified, then get their values from sysgen parameters. If P1 still isn't defined, use FULL.

```

if p1 .nes. "" then goto 9

```

```

$ p1 = f$edit(f$getsysi('STARTUP_P1'),'TRIM,UPCASE')
$ if p1 .eqs. "" then p1 = "FULL"
$ p2 = f$edit(f$getsysi('STARTUP_P2'),'TRIM,UPCASE')
$ p3 = f$edit(f$getsysi('STARTUP_P3'),'TRIM,UPCASE')
$ p4 = f$edit(f$getsysi('STARTUP_P4'),'TRIM,UPCASE')
$ p5 = f$edit(f$getsysi('STARTUP_P5'),'TRIM,UPCASE')
$ p6 = f$edit(f$getsysi('STARTUP_P6'),'TRIM,UPCASE')
$ p7 = f$edit(f$getsysi('STARTUP_P7'),'TRIM,UPCASE')
$ p8 = f$edit(f$getsysi('STARTUP_P8'),'TRIM,UPCASE')
$9:
$!
$! P1 specifies a request code, which tells us what to do. Make sure
$! the code is valid, and expand it to its full spelling.
$
$ run_exit = 'EXIT_OK'
$! **JNL** $ requests = "CACHE_SERVER,CONFIGURE,CSP,ERRFMT,FULL,JOBCTL,MINIMUM,OPCOM,RCP,UPGRADE"
$ requests = "CACHE_SERVER,CONFIGURE,CSP,ERRFMT,FULL,JOBCTL,MINIMUM,OPCOM,UPGRADE"
$ i = f$loc(", "+p1,requests)
$ if i .eq. f$len(requests) then say "%STARTUP-F-INVREQ, The parameter 'p1 is invalid."
$ if i .eq. f$len(requests) then goto EXIT_FATAL
$ p1 = f$ext(i+1,f$loc(", ",f$ext(i+1,999,requests)),requests)
$!
$! P2 determines whether or not we enable verification.
$
$ if p2 then set verify
$
$! Case on the request code.
$
$ goto 'p1

```



```

$FULL:
$MINIMUM:
$UPGRADE:
$ if microvax .and. (P1 .eqs. '') then say -
  $fao('!/The MicroVMS system is now executing the system startup procedure.!/')
$
$! At this point, the following logical names have been defined:
$! SYSSYSDEVICE, SYS$TOPSYS, SYSSCOMMON, SYSSYSROOT, SYSSSYSTEM,
$! SYSSSHARE, and SYSSMESSAGE. Set up SYSSSPECIFIC and redefine SYSSYSROOT
$! if there is no common root. Equate SYSSYSDISK and SYSSYSROOT.
$
$ root = $getdvi('SYSSYSDEVICE','fulldevnam') - ''
$ if $trnlnm('SYS$TOPSYS','LNMSYSTEM') .nes. '' then -
$   root = root + '[' + $trnlnm('SYS$TOPSYS','LNMSYSTEM') + ']'
$ define/system/exec/nolog/translation=(terminal,concealed) sys$specific 'root
$ no_common_root = $search('SYSSSPECIFIC:[000000]SYSCOMMON.DIR') .eqs. ''
$ if no_common_root then -
$   define/system/exec/nolog/translation=(terminal,concealed) sys$sysroot 'root
$ if no_common_root then -
$   define/system/exec/nolog/translation=(terminal,concealed) sys$common 'root
$ define/system/exec/nolog sys$sysdisk sys$sysroot:
$
$! Set up the standard bunch of logical names for each system directory.
$
$ define/system/exec/nolog sys$errorlog sys$sysroot:[syserr]
$ define/system/exec/nolog sys$examples sys$sysroot:[syshlp.examples]
$ define/system/exec/nolog sys$help sys$sysroot:[syshlp]
$ define/system/exec/nolog sys$instruction sys$sysroot:[syscbi]
$ define/system/exec/nolog sys$library sys$sysroot:[syslib]
$ define/system/exec/nolog sys$maintenance sys$sysroot:[sysmaint]
$ define/system/exec/nolog sys$manager sys$sysroot:[sysmgr]
$ define/system/exec/nolog sys$update sys$sysroot:[sysupd]
$ define/system/exec/nolog sys$test sys$sysroot:[systest]
$
$! Define the system logical names used by the Debugger.
$
$ define/system/nolog dbg$input sys$input:
$ define/system/nolog dbg$output sys$output:
$
$! Define the logical name used by DCL as the logical name table name
$! during translates.
$
$ define/table=lnm$system_directory/super lnm$dcl_logical lnm$file_dev
$
$! If a separate system parameter file is being used and modifications
$! were made during SYSBOOT then invoke sysgen to update the system
$! parameter file.
$
$ if $getsysi('WRITESYSPARAMS') then sysgen write current
$
$ if p1 .eqs. 'UPGRADE' then goto POST_INSTALL
$
$! Now we can install a more-or-less standard set of VMS images.
$
$ if $getsysi('TAILORED') then @sys$manager:vmsimages INSTALL vmsimages
$ if .not. $getsysi('TAILORED') then -
$   define /user sys$input sys$manager:vmsimages.dat
$   if .not. $getsysi('TAILORED') then run sys$system:install
$   @sys$manager:vmsimages INSTALL sys$manager:syimages
$
$! POST_INSTALL:
$! Start up the error formatter, cluster services, operator
$! communication, and job controller processes, in that order.
$

```

```
$ run_exit = 'FULL_20'
$ERRFMT:
$ if microvax .and. (p1 .nes. 'ERRFMT') then goto 'run_exit'
$ define/user sys$output nl:
$ run sys$system:errfmt -
    /ast_limit=64 /buffer_limit=40960 -
    /input=opa0: /output=opa0: /error=sys$manager:errfmt_error.log -
    /dump /enqueue_limit=100 /io_buffered=12 /io_direct=12 -
    /maximum_working_set=512 /file_limit=64 -
    /page_file=20480 /priority=7 /queue_limit=64 /subprocess_limit=64 -
    /privileges=(nosame,bypass,cmkrnl,world) -
    /process_name='ERRFMT' /uic=[1,6] /working_set=100
$ goto 'run_exit'
```



```
$full_20:
$   run_exit = "FULL_30"
$   if p1 .eqs. "MINIMUM" then goto 'run_exit'
$CACHE_SERVER:
$   if .not. f$getsysi("CLUSTER_MEMBER") then goto 'run_exit'
$   define/user sys$output nl:
$   run sys$system:fileserv -
      /input=opa0: /output=opa0: /error=sys$manager:cache_server_error.log -
      /ast_limit=64 /buffer_limit=40960 /dump -
      /enqueue_limit=100 /file_limit=64 /io_buffered=12 /io_direct=12 -
      /maximum_working_set=512 /page_file=20480 -
      /priority=16 /privileges=(all) /queue_limit=64 /subprocess_limit=64 -
      /process_name='CACHE_SERVER' /uic=[1,4] /working_set=100
$   goto 'run_exit'
```

```
$full_30:
$   run_exit = 'FULL 40'
$   if p1 .eqs. 'MINIMUM' then goto 'run_exit'
$CSP:
$   if .not. f$getsysi('CLUSTER_MEMBER') then goto 'run_exit'
$   define/user sys$output nl:
$   run sys$system:csp -
      /input= opa0: /output= opa0: /error=sys$manager:cluster_server_error.log -
      /ast_limit=64 /buffer_limit=40960 /dump -
      /enqueue_limit=100 /file_limit=64 /io_buffered=12 /io_direct=12 -
      /maximum_working_set=512 /page_file=20480 -
      /priority=8 /privileges=(all) /queue_limit=64 /subprocess_limit=64 -
      /process_name='CLUSTER_SERVER' /uic=[1,4] /working_set=100
$   goto 'run_exit'
```



```
$full_40:
$ run_exit = 'FULL_60'
$OPCOM:
$ if microvax .and. (p1 .nes. 'OPCOM') then goto 'run_exit'
$ define/user sys$output nl:
$ run sys$system:opcom -
    /input=opa0: /output=opa0: /error=sys$manager:opcom_error.log -
    /ast_limit=96 /buffer_limit=96000 /dump -
    /enqueue_limit=100 /file_limit=64 /io_buffered=64 /io_direct=16 -
    /maximum_working_set=512 /extent=2048 /page_file=20480 /priority=6 -
    /privileges=(nosame,cmkrnl,exquota,oper,sysprv,world,netmbx,setprv) -
    /queue_limit=64 /subprocess_limit=64 -
    /process_name='OPCOM' /uic=[1,4] /working_set=100
$ goto 'run_exit'
```

```
$full_60:
$  run_exit = "FULL_80"
$JOBCTL:
$
$  define/user sys$output nl:
$  run sys$system:jobctl -
    /input=_opa0: /output=_opa0: /error=sys$manager:job_control_error.log -
    /ast_limit=200 /buffer_limit=100000 /dump -
    /enqueue_limit=2000 /file_limit=200 /io_buffered=200 /io_direct=200 -
    /maximum_working_set=512 /page_file=20480 /priority=8 -
    /privileges=(nosame,setprv) - ! Enables its own privileges.
    /queue_limit=200 /subprocess_limit=200 -
    /process_name="JOB_CONTROL" /jic=[1,4] /working_set=100
$  goto 'run_exit'
```


full_80:

```

$! Set console terminal characteristics to guard against site-specific
$! TTY_DEFCHAR modifications.
$!
$! set terminal OPA0 /permanent /nomodem /nodisconnect /nodialup
$!
$! Case again on the request code because the paths diverge now.
$!
$! goto 'p1'_CONTINUE
$
$FULL_CONTINUE:
$!
$! We continue a full startup by configuring the I/O devices.
$! If the user has a configuration procedure, we invoke it.
$! Then, unless told otherwise, we autoconfigure all devices.
$
$ startup$autoconfigure_all == 1
$ if f$search('sys$manager:syconfig.com') .nes. '' then =
$   @sys$manager:syconfig 'p1' 'p2' 'p3' 'p4' 'p5' 'p6' 'p7' 'p8'
$ if startup$autoconfigure_all .and. .not. f$getsyi('NOAUTOCONFIG') then -
$   sysgen autoconfigure all
$
$! After the devices are configured, we can start up the dynamic
$! configuration process.
$
$ if .not. startup$autoconfigure_all .or. f$getsyi('NOAUTOCONFIG') -
$   then goto configure_started
$ run_exit = 'configure_started'
$CONFIGURE:
$ if .not. f$getdvi('PAA0:', 'EXISTS') then goto 'run_exit'
$ define/user sys$output nl:
$ run sys$system:configure -
$   /input= opa0: /output= opa0: /error=sys$manager:configure_error.log -
$   /ast_limit=200 /buffer_limit=100000 /dump -
$   /enqueue_limit=200 /file_limit=200 /io_buffered=200 /io_direct=200 -
$   /maximum_working_set=512 /page_file=20480 /priority=8 -
$   /privileges=(nosame,cmkrnl,prmmmbx,bypass,share) -
$   /queue_limit=200 /subprocess_limit=200 -
$   /process_name='CONFIGURE' /uic=[1,4] /working_set=100
$ goto 'run_exit'

```

```

$configure_started:
$
$!      Install any deferred swapfile if it exists.
$
$      if f$search('sys$system:swapfile1.sys') .nes. '' then -
$          sysgen install sys$system:swapfile1.sys /swapfile
$
$!      Invoke UVSTARTUP if this is a MicroVMS system
$
$      if microvax .and. (f$search('sys$system:uvstartup.com')) .nes. '' then -
$          @sys$system:uvstartup
$!      If the system crashed during a software product installation performed
$!      by VMSINSTAL, then we want to invoke VMSINSTAL to attempt recovery.
$!      The existence of a marker file gives us the clue we need.
$
$      if f$search('sys$update:vmimarker.dat') .nes. '' then -
$          @sys$update:vmsinstal '""' options b
$
$! **JNL** $!      Start Common Journaling if SYSGEN parameter demands it, or if in
$! **JNL** $!      cluster.
$! **JNL** $
$! **JNL** $      if f$getsyi('CLUSTER_MEMBER') then goto cjffload
$! **JNL** $
$! **JNL** $!      If CJF isn't wanted, we're done.
$! **JNL** $
$! **JNL** $      if .not. f$getsyi('CJFLOAD') then goto cjfdone
$! **JNL** $
$! **JNL** $cjffload:
$! **JNL** $!
$! **JNL** $!      Start the default journal ACP
$! **JNL** $!
$! **JNL** $      run sys$system:jcp
$! **JNL** $start
$! **JNL** $exit
$! **JNL** $!      Load the journaling driver and create the UCBs for BI, AI, AT, RU, CL journals.
$! **JNL** $
$! **JNL** $      run sys$system:sysgen
$! **JNL** $load jnldriver
$! **JNL** $connect cjb0/drivername=jnldriver/noadapter
$! **JNL** $connect cja0/drivername=jnldriver/noadapter
$! **JNL** $connect cjt0/drivername=jnldriver/noadapter
$! **JNL** $connect cjr0/drivername=jnldriver/noadapter
$! **JNL** $connect cjc0/drivername=jnldriver/noadapter
$! **JNL** $exit
$! **JNL** $      define/system CJF$DRV_LOADED y
$! **JNL** $
$! **JNL** $!      Load the CJF and RUF system services
$! **JNL** $
$! **JNL** $      run sys$system:sysgen
$! **JNL** $load sys$system:cjfla.exe
$! **JNL** $load sys$system:rufloa.exe
$! **JNL** $exit
$! **JNL** $      define/system CJF$SERV_LOADED y
$! **JNL** $!
$! **JNL** $!      Start the Recovery Control Process
$! **JNL** $!
$! **JNL** $      run_exit = 'RCP_LOADED'
$! **JNL** $RCP:
$! **JNL** $      run sys$system:rcp.exe -
$! **JNL** $          /input="" /output="" /error="SYS$MANAGER:JNLRCP_ERROR.LOG" -
$! **JNL** $          /process_name="JNLRCP" /utc=[1,3] /priority=4 /ast_limit=200 -
$! **JNL** $          /buffer_limit=100000 /enqueue_limit=200 /io_buffered=200 /io_direct=200 -
$! **JNL** $          /file_limit=200 /working_set=T00 /maximum_working_set=512 /page_file=20480 -

```



```

C 9
$! **JNL** /queue_limit=200 /subprocess_limit=200 -
$! **JNL** /privileges=(nosame,noacct,altpri,cmkrnl,detach,exquota,log_io,setprv,tmpmbx,prmbx,sysnam,world,share)
$! **JNL** $ goto 'run_exit'
$! **JNL** $rcp_loaded:
$! **JNL** $
$! **JNL** $ If in a cluster, solicit information about other nodes in the cluster.
$! **JNL** $
$! **JNL** $ run sys$system:jcp
$! **JNL** initialize
$! **JNL** exit
$! **JNL** $ define/system CJF$INITIALIZED y
$! **JNL** $
$! **JNL** $ If a Recovery Unit Journal is needed on the system disk, create it.
$! **JNL** $ If other than the default parameters imposed by JCP are desired, the
$! **JNL** $ correct approach is:
$! **JNL** $ 1. bring the system up with CJFSYSRUJ = 0,
$! **JNL** $ 2. manually create the recovery unit journal with the desired characteristics,
$! **JNL** $ 3. set CJFSYSRUJ = 1 for subsequent boots.
$! **JNL** $
$! **JNL** $ if .not. f$getsyi("CJFSYSRUJ") then goto nosysruj
$! **JNL** $ run sys$system:jcp
$! **JNL** create/file=cif rujnl ru sys$sysdevice:
$! **JNL** $
$! **JNL** $ That's about it.
$! **JNL** $
$! **JNL** $nosysruj:
$! **JNL** $scjfdone:
$
$ Set up a 'normal' environment and invoke the site-specific
$ startup procedure, if present.
$
$ set default sys$system
$
$ Set up the global section needed by the RTL SMG$ facility
$
$ run sys$system:smgmaptrm
$
$ define/nolog sys$login sys$system:
$ define/job/nolog sys$scratch sys$manager:
$ mount := mount/noassist
$ startup$interactive_logins == 64
$ if f$search('sys$manager:systartup.com') .nes. "" then =
$ @sys$manager:systartup ""p1"" ""p2"" ""p3"" ""p4"" ""p5"" ""p6"" ""p7"" ""p8""
$
$ In V3, the site-specific startup procedure was not supposed to return
$ unless it wanted 64 interactive logins. Now we use the global symbol
$ to determine the login count. But since a site-specific procedure
$ might not return, we can't do anything after this point except enable
$ logins and quit.
$
$ set logins/interactive='startup$interactive_logins
$ logout/brief

```


\$MINIMUM_CONTINUE:

```

$
$
$! Enable eight interactive logins and call it quits.
$

```

```
$ set logins/interactive=8
$ logout/brief
```

$\frac{f}{F}$

\$UPGRADE_CONTINUE:

\$
\$! We are done with a startup for the upgrade phases. However, we
\$! want to exit rather than logging out.
\$
\$ goto EXIT_OK

F 9

```
$! We always come down here to exit the procedure, so that verification
$! can be reset as appropriate.
$
$EXIT_OK:
$   set default 'saved_default
$   set message 'saved_msg
$   exit %x103a0001 + 0*'if$verify(saved_verify)
$
$EXIT_FATAL:
$   set default 'saved_default
$   set message 'saved_msg
$   exit %x103a0004 + 0*'if$verify(saved_verify)
```


